GmbH Informationssysteme Engineering & Consulting
Technoparkstrasse 1
8005 Zürich
Tel.: 044 / 350 10 10
Fax.: 044 / 350 10 19

GeoShop SOAP Benutzerhandbuch

Zusammenfassung

Diese Dokumentation beschreibt die GeoShop SOAP-Schnittstelle.

Inhaltsverzeichnis

1. Einleitung

1.1. Überblick

Die **GeoShop SOAP Schnittstelle** ist eine WSDL/SOAP [1] basierte Schnittstelle für die Kommuninaktion mit dem GeoShop. So könnem mit der Schnittstelle zum Beispiel Produkte bestellt werden. Somit können GeoShop Dienste direkt in Applikationen von Drittanbietern integriert werden.

1.2. Beispiele

Beispiele für die Anwendung der GeoShop Soap Methoden sind im GeoShop unter `geoshop\client\soap` zu finden. Diese Beipiele werden mit `Curl` angewendet.

1.3. Aufbau dieser Dokumentation

Diese Dokumentation ist in folgende Kapitel gegliedert:

- Kapitel 2: UserService für Methoden zu einem GeoShop User.
- Kapitel 3: OrderService für Methoden für GeoShop Bestellungen.
- Kapitel 4: TransformService für Methoden für Geometrien transformieren.
- Kapitel 5: Beispiel Integration Services in eine Applikation.

1.4. Ergänzende Dokumentationen

[1] Simple Object Access Protocol (SOAP), s.a. `http://www.w3.org/TR/SOAP`.

[2] GeoShop BatchClient Benutzerhandbuch. Im GeoShop BatchClient Benutzerhandbuch ist eine Beschreibung aller möglichen Bestellparameter enthalten.

1.5. Konventionen

In dieser Dokumentation werden folgende Konventionen eingehalten:

`Kursiv`	Namen von Dateien und URL's
fett	neue Begriffe, Namen von Funktionen oder Methoden
`courier`	Programmtext oder Eingaben im Betriebssystem

2. UserService

2.1. Einleitung

Mit dem **GeoShop UserService** können Informationen zu einem GeoShop User abgefragt werden wie zum Beispiel die Produkte des Users.

2.2. UserService Endpunkte und Methoden

2.2.1. Service Endpunke

Der GeoShop UserService kann via folgende SOAP bzw. WSDL Service Endpunkte erreicht werden:

`GEOSHOP_URL`/soap/UserService.igs
: SOAP Schnittstelle.

`GEOSHOP_URL`/soap/UserService.igs?wsdl
: WSDL Beschreibung der SOAP Schnittstelle.

Wobei für `GEOSHOP_URL` die Basisadresse des GeoShop eingesetzt werden muss (z.B. `http://localhost:3501`).

2.2.2. getUser Methode

2.2.2.1. Beschreibung

Mit `getUser` Methode wird geprüft, ob der User im GeoShop definiert ist.

2.2.2.2. XML Input Objekt

Die `getUser` Methode wird durch folgendes XML-Objekt aufgerufen (Darstellung mit `soap:Body` und `soap:Envelope`):

```
<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope
   xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"
   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
   xmlns:xsd="http://www.w3.org/2001/XMLSchema">
<soap:Body>

<getUser xmlns="http://www.infogrips.ch/geoshop/user/">
   <user xmlns="">user</user>
   <password xmlns="">password</password>
</getUser>

</soap:Body>
</soap:Envelope>
```

Beschreibung der variablen Anteile:

user (String)

GeoShop Benutzer.

password (String)

Passwort des GeoShop Benutzers.

Der Parameter `server` wird automatisch vom GeoShop gesetzt und muss daher nicht übermittelt werden. Für die obligatorischen Parameter `user`, `password` müssen die dafür reservierten Tags (s.a. oben) benutzt werden.

2.2.2.3. XML Output Objekt

Die `getUser` Methode liefert als Resultat der Anfrage folgendes XML-Objekt (Darstellung mit `soap:Body` und `soap:Envelope`):

```
<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope
    xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xmlns:xsd="http://www.w3.org/2001/XMLSchema">
<soap:Body>

<getUserResponse xmlns="http://www.infogrips.ch/geoshop/user/">
</getUserResponse>

</soap:Body>
</soap:Envelope>
```

Es werden keine Variablen Anteile zurückgegeben. Ist der User oder das Password ungültig, ist der Request auch ungültig.

2.2.3. getProducts Methode

2.2.3.1. Beschreibung

Die `getProducts` Methode frägt die Produkte eines Users ab.

2.2.3.2. XML Input Objekt

Die `getProducts` Methode wird durch folgendes XML-Objekt aufgerufen (Darstellung mit `soap:Body` und `soap:Envelope`):

```
<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope
   xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"
   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
   xmlns:xsd="http://www.w3.org/2001/XMLSchema">
<soap:Body>

<getProducts xmlns="http://www.infogrips.ch/geoshop/user/">
    <user xmlns="">test</user>
    <password xmlns="">test</password>
</getProducts>
```

```
</soap:Body>
</soap:Envelope>
```

Beschreibung der variablen Anteile:

user (String)
: GeoShop Benutzer.

password (String)
: Passwort des GeoShop Benutzers.

Der Parameter `server` wird automatisch vom GeoShop gesetzt und muss daher nicht übermittelt werden. Für die obligatorischen Parameter `user`, `password` müssen die dafür reservierten Tags (s.a. oben) benutzt werden.

2.2.3.3. XML Output Objekt

Die `getProducts` Methode liefert als Resultat der Anfrage folgendes XML-Objekt (Darstellung mit `soap:Body` und `soap:Envelope`):

```
<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope
   xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"
   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
   xmlns:xsd="http://www.w3.org/2001/XMLSchema">
<soap:Body>

<getProductsResponse xmlns="http://www.infogrips.ch/geoshop/user/">
   <products xmlns="">
      <product xmlns="">
         <name>dxf_dm01</name>
         <display_name>AV DXF/DWG</display_name>
      </product>
      <product xmlns="">
         <name>dxf_plot</name>
         <display_name>AV DXF Plot</display_name>
      </product>
      :
   </products>
</getProductsResponse>

</soap:Body>
</soap:Envelope>
```

Beschreibung der variablen Anteile:

products (list)
: Eine Liste von Produkten.

product (Struktur)
: Ein Produkt.

product.name (String)
: Der Name des Produktes.

product.display_name (String)
: Der Displayname des Produktes.

2.2.4. getProductDefinition Methode

2.2.4.1. Beschreibung

Die `getProductDefinition` Methode frägt die Definition eines Produktes eines Users ab.

2.2.4.2. XML Input Objekt

Die `getProductDefinition` Methode wird durch folgendes XML-Objekt aufgerufen (Darstellung mit `soap:Body` und `soap:Envelope`):

```
<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope
   xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"
   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
   xmlns:xsd="http://www.w3.org/2001/XMLSchema">
<soap:Body>

<getProductDefinition xmlns="http://www.infogrips.ch/geoshop/user/">
   <user xmlns="">test</user>
   <password xmlns="">test</password>
   <product xmlns="">pdf</product>
</getProductDefinition>

</soap:Body>
</soap:Envelope>
```

Beschreibung der variablen Anteile:

user (String)
: GeoShop Benutzer.

password (String)
: Passwort des GeoShop Benutzers.

product (String)
: Name des Produktes.

Der Parameter `server` wird automatisch vom GeoShop gesetzt und muss daher nicht übermittelt werden. Für die obligatorischen Parameter `user`, `password`, `product` müssen die dafür reservierten Tags (s.a. oben) benutzt werden.

2.2.4.3. XML Output Objekt

Die `getProductDefinition` Methode liefert als Resultat der Anfrage folgendes XML-Objekt (Darstellung mit `soap:Body` und `soap:Envelope`):

```
<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope
   xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"
   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
   xmlns:xsd="http://www.w3.org/2001/XMLSchema">
<soap:Body>

   <getProductDefinitionResponse xmlns="http://www.infogrips.ch/geoshop/user/">
      <productdefinition xmlns="">
```

```
         <name>pdf</name>
         <display_name>AV Adobe PDF</display_name>
         <models xmlns="">
            <model xmlns="">
               <name>DM01AVCH24D</name>
               <display_name>amtl. Vermessung</display_name>
               <topics xmlns="">
                  <topic>FixpunkteKategorie1</topic>
                  <topic>FixpunkteKategorie2</topic>
                  :
               </topics>
            </model>
         </models>
         <params xmlns="">
            <selection_type>FORMATBOX</selection_type>
            <selection_formats xmlns="">
               <selection_format xmlns="">
                  <format>A4</format>
                  <orientations>hoch,quer</orientations>
                  <scales>1:250,1:500,1:1000</scales>
                  <format_default>ON</format_default>
                  <orientation_default>hoch</orientation_default>
                  <scale_default>1:250</scale_default>
               </selection_format>
               :
            </selection_formats>
            <selection_options xmlns="">
               <option xmlns="">
                  <name>coordband</name>
                  <value>checkbox,Koordinatenband,on</value>
               </option>
               <option xmlns="">
                  <name>coordcross</name>
                  <value>checkbox,Koordinatenkreuze,on</value>
               </option>
               :
            </selection_options>
            <topic_selection_visible>OFF</topic_selection_visible>
            <order_la_options xmlns="">
               <option xmlns="">
                  <name>name1</name>
                  <value>textfield,Name1 *</value>
               </option>
               <option xmlns="">
                  <name>name2</name>
                  <value>textfield,Name2 *</value>
               </option>
               :
            </order_la_options>
            <option_ra_option>ON</option_ra_option>
         </params>
         <price_function>ON</price_function>
      </productdefinition>
   <getProductDefinitionResponse>

</soap:Body>
</soap:Envelope>
```


Beschreibung der variablen Anteile:

productdefinition (Struktur) *

Eine Struktur, die das Produkt definiert.

productdefinition.name (String) *

Der Name des Produktes.

productdefinition.display_name (String) *

Der Displayname des Produktes.

productdefinition.models (Liste) *

Eine Liste mit allen Modellen, die im Produkt enthalten sein können.

productdefinition.models.model (Struktur) *

Ein Modell, das im Produkt enthalten sein kann.

productdefinition.models.model.name (String) *

Name des Modelles.

productdefinition.models.model.display_name (String)

Displayname des Modelles

productdefinition.models.model.topics (List) *

Eine Liste der Topics des Modelles.

productdefinition.models.model.topics.topic (String) *

Name des Topics.

productdefinition.params (Struktur) *

Eine Struktur mit weitere Parametern zum Produkt.

productdefinition.params.selection_type (String)

Definiert die geographische Selektion für das Produkt. Mögliche Werte sind: BOX (default),POLYGON,FORMATBOX.

productdefinition.params.selection_rotate (String)

Definiert, ob die geographische Selektion rotiert werden kann. Mögliche Werte sind: ON (default),OFF

productdefinition.params.selection_formats (Liste)

Liste von Formaten für `selection_type=FORMATBOX`.

productdefinition.params.selection_formats.selection_format (Struktur)

Format für `selection_type=FORMATBOX`.

productdefinition.params.selection_options (Liste)

Weitere Optionen für das Produkt. Abhängig vom Produkt.

productdefinition.params.selection_options.option (Struktur)

Option für das Produkt. Abhängig vom Produkt.

productdefinition.params.topic_selection_visible (String)

Kann der User die Modells/Topics selektieren. Mögliche Werte sind: ON (default),OFF

productdefinition.params.order_ra_option (String)

Können Informationen zur Rechnungsadresse eingegeben werden. Mögliche Werte sind: ON,OFF (default).

productdefinition.params.order_la_options (Liste)

Weitere Optionen für die Lieferadresse.

productdefinition.params.order_la_options.option (Struktur)

Option zur Lieferadresse.

productdefinition.params.order_ra_options (Liste)

Weitere Optionen für die Rechnungsadresse.

productdefinition.params.order_ra_options.option (Struktur)

Option zur Rechnungsadresse.

productdefinition.proce_function (String)

Hat das Produkt eine Preisfunktion. Mögliche Werte sind: ON,OFF (default).

Die mit einem * markierten Anteile sind immer vorhanden. Die anderen Anteile können je nach Produkt vorhanden sein.

2.2.5. getPreferences Methode

2.2.5.1. Beschreibung

Die `getPreferences` Methode frägt die Präferenzen eines Users ab.

2.2.5.2. XML Input Objekt

Die `getPreferences` Methode wird durch folgendes XML-Objekt aufgerufen (Darstellung mit `soap:Body` und `soap:Envelope`):

```
<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope
   xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"
   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
   xmlns:xsd="http://www.w3.org/2001/XMLSchema">
<soap:Body>

<getPreferences xmlns="http://www.infogrips.ch/geoshop/user/">
   <user xmlns="">test</user>
   <password xmlns="">test</password>
</getPreferences>

</soap:Body>
</soap:Envelope>
```

Beschreibung der variablen Anteile:

user (String)

GeoShop Benutzer.

password (String)

Passwort des GeoShop Benutzers.

Der Parameter `server` wird automatisch vom GeoShop gesetzt und muss daher nicht übermittelt werden. Für die obligatorischen Parameter `user`, `password` müssen die dafür reservierten Tags (s.a. oben) benutzt werden.

2.2.5.3. XML Output Objekt

Die `getPreferences` Methode liefert als Resultat der Anfrage folgendes XML-Objekt (Darstellung mit `soap:Body` und `soap:Envelope`):


```
<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope
    xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xmlns:xsd="http://www.w3.org/2001/XMLSchema">
<soap:Body>
   <getPreferencesResponse xmlns="http://www.infogrips.ch/geoshop/user/">
      <preferences xmlns="">
         <order.adr1></order.adr1>
         <order.adr2></order.adr2>
         <order.city></order.city>
         <order.country></order.country>
         <order.email></order.email>
         <order.fax></order.fax>
         <order.name1></order.name1>
         <order.name2></order.name2>
         <order.remark></order.remark>
         <order.tel></order.tel>
         <order.zip></order.zip>
         <overview.maxX>678600.0</overview.maxX>
         <overview.maxY>247600.0</overview.maxY>
         <overview.minX>672900.0</overview.minX>
         <overview.minY>242600.0</overview.minY>
         <overview.view>overview</overview.view>
         <range.maxX>675859.761555</range.maxX>
         <range.maxY>245435.0</range.maxY>
         <range.minX>675766.238445</range.minX>
         <range.minY>245364.0</range.minY>
         <search.query>strasse</search.query>
      </preferences>
   </getPreferencesResponse>
</soap:Body>
</soap:Envelope>
```

Beschreibung der variablen Anteile:

order.name1 (String)
: Namenfeld 1.

order.name2 (String)
: Namenfeld 2.

order.adr1 (String)
: Adresse 1.

order.adr2 (String)
: Adresse 2.

order.zip (String)
: Postleitzahl.

order.city (String)
: Stadt.

order.country (String)
: Land.

order.tel (String)
: Telefon.

order.fax (String)

Fax..

order.email (String)

EMail-Addresse.

Dies sind die üblichen Anteile für Users die Bestellungen ausführen können. Es können weitere Anteile vorhanden sein.

Grundsätzlich sind alle Anteile optional.

3. OrderService

3.1. Einleitung

Mit dem **GeoShop OrderService** kann eine Preisberechung und Bestellung von Daten von einem GeoShop Server durchgeführt werden. Der GeoShop OrderService hat den gleichen Funktionsumfang wie die Applikation **GeoShop BatchClient** (s.a. [2]). Der GeoShop Batch-Client kann daher alternativ zum GeoShop OrderService verwendet werden.

3.2. OrderService Endpunkte und Methoden

3.2.1. Service Endpunke

Der GeoShop OrderService kann via folgende SOAP bzw. WSDL Service Endpunkte erreicht werden:

`GEOSHOP_URL`/soap/orderservice.igs
: SOAP Schnittstelle.

`GEOSHOP_URL`/soap/orderservice.igs?wsdl
: WSDL Beschreibung der SOAP Schnittstelle.

Wobei für `GEOSHOP_URL` die Basisadresse des GeoShop eingesetzt werden muss (z.B. `http://localhost:3501`).

3.2.2. calculatePrice Methode

3.2.2.1. Beschreibung

Die `caculatePrice` Methode liefert das Resultat der GeoShop Preisberechnung. Neben dem Preis (Betrag/Währung) wird auch ein formatierter Beleg bzw. eine formatierte Fehlermeldung zurück geliefert. Der Beleg bzw. die Fehlermeldung entspricht der Ausgabe der Preisberechnung im GeoShop Client Applet.

3.2.2.2. XML Input Objekt

Die `calculatePrice` Methode wird durch folgendes XML-Objekt aufgerufen (Darstellung mit `soap:Body` und `soap:Envelope`):

```
<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope
   xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"
   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
   xmlns:xsd="http://www.w3.org/2001/XMLSchema">
<soap:Body>

<calculatePrice xmlns="http://www.infogrips.ch/geoshop/order/">
   <user xmlns="">user</user>
   <password xmlns="">password</password>
   <product xmlns="">product</product>
```

```
   <parameters xmlns="">
      <parameter>
         <name>name</name>
         <value>value</value>
      </parameter>
      ...
   </parameters>
</calculatePrice>

</soap:Body>
</soap:Envelope>
```

Beschreibung der variablen Anteile:

user (String)

GeoShop Benutzer.

password (String)

Passwort des GeoShop Benutzers.

product (String)

Datenprodukt für welches der Preis berechnet werden soll (muss dem Benutzer zugewiesen sein).

parameters (List of <parameter>)

Liste von 0-N Parametern. Die Anzahl der notwendigen Parameter ist abhängig vom gewählten Datenprodukt. Die Parameter sind in [2] beschrieben. Beispiel:

```
<parameters>
   <parameter>
      <name>min</name>
      <value>600000/200000</value>
   </parameter>
   <parameter>
      <name>max</name>
      <value>620000/250000</value>
   </parameter>
</paramaters>
```

Der Parameter `server` wird automatisch vom GeoShop gesetzt und muss daher nicht übermittelt werden. Für die obligatorischen Parameter `user`, `password` und `product` müssen die dafür reservierten Tags (s.a. oben) benutzt werden.

3.2.2.3. XML Output Objekt

Die calculatePrice Methode liefert als Resultat der Anfrage folgendes XML-Objekt (Darstellung mit `soap:Body` und `soap:Envelope`):

```
<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope
   xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"
   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
   xmlns:xsd="http://www.w3.org/2001/XMLSchema">
<soap:Body>

<calculatePriceResponse xmlns="http://www.infogrips.ch/geoshop/order/">
   <priceinfo xmlns="">
      <item>price</item>
      <item>currency</item>
```

```
      <item>1. line of price display</item>
      <item>2. line of price display</item>
      ...
      <item>n. line of price display</item>
   </priceinfo>
</calculatePriceResponse>

</soap:Body>
</soap:Envelope>
```

Beschreibung der variablen Anteile:

priceinfo (List of String)

Liste von Stringwerten, wobei die ersten beiden Werte der Liste eine spezielle Bedeutung haben.

price (Double)

Preis der Bestellung. Ein negativer Preis bedeutet, dass die Bestellung nicht ausgeführt werden kann.

currency (String)

Währung.

weitere Zeilen (String)

Formatierter Beleg bzw. Fehlermeldung (bei negativem Preis).

3.2.3. sendOrder Methode

3.2.3.1. Beschreibung

Die `sendOrder` Methode schickt eine Bestellung an den GeoShop. Die `sendOrder` Methode wird normalerweise asynchron ausgeführt, d.h. es wird nicht auf die Fertigstellung der Bearbeitung gewartet (die Benachrichtigung des Benutzers erfolgt per E-Mail). Es ist jedoch auch möglich die `sendOrder` Methode synchron auszuführen. Dazu muss der Bestellparameter `wait` wie folgt gesetzt werden:

```
<parameters>
   ...
   <parameter>
      <name>wait</name>
      <value></value>
   </parameter>
   ...
</paramaters>
```

3.2.3.2. XML Input Objekt

Die `sendOrder` Methode wird durch folgendes XML-Objekt aufgerufen (Darstellung mit `soap:Body` und `soap:Envelope`):

```
<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope
   xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"
   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
   xmlns:xsd="http://www.w3.org/2001/XMLSchema">
<soap:Body>
```

```
<sendOrder xmlns="http://www.infogrips.ch/geoshop/order/">
   <user xmlns="">user</user>
   <password xmlns="">password</password>
   <product xmlns="">product</product>
   <parameters xmlns="">
      <parameter>
         <name>name</name>
         <value>value</value>
      </parameter>
      ...
   </parameters>
</sendOrder>

</soap:Body>
</soap:Envelope>
```

Beschreibung der variablen Anteile, s.a. `calculatePrice`.

3.2.3.3. XML Output Objekt

Die `sendOrder` Methode liefert als Resultat der Anfrage folgendes XML-Objekt (Darstellung mit `soap:Body` und `soap:Envelope`):

```
<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope
   xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"
   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
   xmlns:xsd="http://www.w3.org/2001/XMLSchema">
<soap:Body>

<sendOrderResponse xmlns="http://www.infogrips.ch/geoshop/order/">
   <order xmlns="">
     <orderno>orderno</orderno>
     <orderurl>orderurl</orderurl>
     <price>price</price>
     <currency>currency</currency>
   </order>
</sendOrderResponse>

</soap:Body>
</soap:Envelope>
```

Beschreibung der variablen Anteile:

orderno (Integer)
: Bestellnummer. Falls die Bestellnummer < 0 ist, konnte die Bestellung nicht verarbeitet werden.

orderurl (String)
: URL an welcher die bestellten Daten verfügbar sind.

price (Double)
: Preis.

currency (String)
: Währung.

4. TransformService

4.1. Einleitung

Mit dem **GeoShop TransformService** kann ein Punkt oder eine andere Geometry von einem Koordinatenreferenzsystem in ein anderes Koordinatenreferenzsystem transformiert werden. Zum Beispiel die Schweizer Koordinaten von LV03 nach LV95.

Ein Anwendungsbeispiel kann eine Datenbestellung beim GeoShop über den OrderService sein. Der Bestellbereich ist in LV03 vorhanden, das Produkt für die Bestellung verlangt aber den Bestellbereich in LV95. Mit dem TransformService kann der Bestellbereich von LV03 nach LV95 transformiert werden.

4.2. TransformService Endpunkte und Methoden

4.2.1. Service Endpunke

Der GeoShop TransformService kann via folgende SOAP bzw. WSDL Service Endpunkte erreicht werden:

`GEOSHOP_URL`/soap/transformservice.igs
: SOAP Schnittstelle.

`GEOSHOP_URL`/soap/transformservice.igs?wsdl
: WSDL Beschreibung der SOAP Schnittstelle.

Wobei für `GEOSHOP_URL` die Basisadresse des GeoShop eingesetzt werden muss (z.B. `http://localhost:3501`).

4.2.2. Koordinatenreferenzsysteme

Für die Transformationen werden die Input- und Output-Koordinatenreferenzsysteme mit dem EPSG-Code angegeben. Beispiele von EPSG-Codes sind:

2056
: CH1903+ / LV95 Schweizer Koordinatenreferenzsystem LV95

3857
: WGS 84 / Pseudo-Mercator, Google, OpenStreetMap, etc.

4326
: WGS 84

21781
: CH1903 / LV03 Schweizer Koordinatenreferenzsystem LV03

21782
: FL1903 / LV03 Lichtensteinisches Koordinatenreferenzsystem LV03

Weitere Koordinatenreferenzsysteme werden bei Bedarf implementiert.

4.2.3. Geometrien

Die Methode `getPoint` verlangt einen Punkt als x,y,z Koordinaten.

Die Methode `getGeometry` verlang eine Geometrie als OGC WKT Geometrie.

Beispiele von OGC WKT Geometrien sind:

```
POINT (30 10)
LINESTRING (30 10, 10 30, 40 40)
POLYGON ((30 10, 40 40, 20 40, 10 20, 30 10))
```

Mehr zu OGC WKT Geometrien siehe unter den Dokumentationen der OGC.

4.2.4. getPoint Methode

4.2.4.1. Beschreibung

Die `getPoint` Methode transformiert einen Punkt mit den Koordinaten `x,y,z` von einem Koordinatenreferenzsystem in ein anderes Koordinatenreferenzsystem.

Die Methode `getGeometry` ist eine allgemeinere Methode und transformiert neben Punkten auch andere Geometrietypen.

4.2.4.2. XML Input Objekt

Die `getPoint` Methode wird durch folgendes XML-Objekt aufgerufen (Darstellung mit `soap:Body` und `soap:Envelope`):

```
<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope
   xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"
   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
   xmlns:xsd="http://www.w3.org/2001/XMLSchema">

   <soap:Body>
      <getPoint xmlns="http://www.infogrips.ch/geoshop/user/">
         <epsgin xmlns="">epsgin</epsgin>
         <epsgout xmlns="">epsgout</epsgout>
         <pointin xmlns="">
            <x xmlns="">x</x>
            <y xmlns="">y</y>
            <z xmlns="">z</z>
         </pointin>
      </getPoint>
   </soap:Body>

</soap:Envelope>
```

Beschreibung der variablen Anteile:

epsgin (Integer)
: EPSG Code des Input Koordinatenreferenzsystemes

epsgout (Integer)
: EPSG Code des Output Koordinatenreferenzsystemes

x (Double)
: Input x Koordinate

y (Double)
: Input y Koordinate

z (Double)
: Input z Koordinate

4.2.4.3. XML Output Objekt

Die `getPoint` Methode liefert als Resultat der Anfrage folgendes XML-Objekt (Darstellung mit `soap:Body` und `soap:Envelope`):

```
<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope
    xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xmlns:xsd="http://www.w3.org/2001/XMLSchema">

   <soap:Body>
    <getPointResponse xmlns="http://www.infogrips.ch/geoshop/user/">
      <pointout xmlns="">
         <x>x</x>
         <y>y</y>
         <z>z</z>
      </pointout>
    </getPointResponse>
  </soap:Body>

</soap:Envelope>
```

Beschreibung der variablen Anteile:

x (Double)
: Output x Koordinate

y (Double)
: Output y Koordinate

z (Double)
: Output z Koordinate

4.2.5. getGeometry Methode

4.2.5.1. Beschreibung

Die `getGeometry` Methode transformiert einen OGC WKT Geometrie von einem Koordinatenreferenzsystem in ein anderes Koordinatenreferenzsystem. DIe OGC WKT Geometrie kann eine von der OGC unterstützen Geometrietypen sein.

4.2.5.2. XML Input Objekt

Die `getGeometry` Methode wird durch folgendes XML-Objekt aufgerufen (Darstellung mit `soap:Body` und `soap:Envelope`):


```
<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope
   xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"
   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
   xmlns:xsd="http://www.w3.org/2001/XMLSchema">

   <soap:Body>
      <getGeometry xmlns="http://www.infogrips.ch/geoshop/user/">
         <epsgin xmlns="">epsgin</epsgin>
         <epsgout xmlns="">epsgout</epsgout>
         <geometryin xmlns="">geometry</geometryin>
       </getGeometry>
   </soap:Body>

</soap:Envelope>
```

Beschreibung der variablen Anteile:

epsgin (Integer)
: EPSG Code des Input Koordinatenreferenzsystemes

epsgout (Integer)
: EPSG Code des Output Koordinatenreferenzsystemes

geometry (String OGC WKT)
: Input Geometrie als OGC WKT Geometrie

4.2.5.3. XML Output Objekt

Die `getGeometry` Methode liefert als Resultat der Anfrage folgendes XML-Objekt (Darstellung mit `soap:Body` und `soap:Envelope`):

```
<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope
    xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xmlns:xsd="http://www.w3.org/2001/XMLSchema">

   <soap:Body>
     <getGeometryResponse xmlns="http://www.infogrips.ch/geoshop/user/">
        <geometryout>geometry</geometryout>
     </getGeometryResponse>
   </soap:Body>

</soap:Envelope>
```

Beschreibung der variablen Anteile:

geometry (String OGC WKT)
: Output Geometrie als OGC WKT Geometrie

5. Beispiel Integration Services in eine Applikation

Nachfolgend wird informell erläutert, wie die GeoShop SOAP Services in eine Applikation integriert werden können, um eine Bestellung auf dem GeoShop auszulösen.

1. **Login**

   Die Applikation fordert den Anwender zu einem Login mit User und Password auf.

   Mit `UserService.getUser` überprüft die Applikation, ob das eingegebene Loging mit User/Password einem gültigen GeoShop User entspricht.

   Wenn ok, weiter mit 2.

   Wenn nicht ok entsprechende Meldung an den Anwender.

2. **Produkt**

   Mit `UserService.getProducts` liest die Applikation, die dem GeoShop zugeordnetet Produkte.

   Die Applikation bietet die Produkte dem Anwender zur Auswahl an. Der Anwender selektiert ein Produkt.

3. **Selektion**

   Mit `UserService.getProductDefinition` liest die Applikation, die Produktdefinition des selektierten Produktes.

   Aufgrund der Produktedefinition lässt die Applikation den Anwender die Produktparameter selektieren, wie zum Beispiel den Bereich, die Modelle und Topics, die Lieferadresse und eventuell weitere Optionen.

4. **Preisfunktion**

   Hat das Produkt eine Preisfunktion, so berechnet die Applikation mit `OrderService.calculatePrice` den Preis und zeigt dem Anwender den Preis zum akzeptieren an.

   Wenn Preis akzeptiert, weiter mit 5.

   Wenn Preis nicht akzeptiert weiter mit 2 oder 3.

5. **Bestellung**

   Mit `OrderService.sendOrder` sendet die Applikation die Bestellung an den GeoShop. Der GeoShop führt die Bestellung aus.

   Als Information zeigt die Applikation dem Anwender die Bestellnummer an.

   Weiter mit 2 oder 3 für eine nächste Bestellung.

A. UserService WSDL Definitionen

```
<?xml version="1.0"?>
<definitions name="UserService"
   targetNamespace="http://www.infogrips.ch/geoshop/user/"
   xmlns:tns="http://www.infogrips.ch/geoshop/user/"
   xmlns:types="http://www.infogrips.ch/geoshop/types/"
   xmlns:xsd="http://www.w3.org/2001/XMLSchema"
   xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
   xmlns="http://schemas.xmlsoap.org/wsdl/">

   <types>
      <xsd:schema targetNamespace="http://www.infogrips.ch/geoshop/types/">

         <xsd:complexType name="StringList">
            <xsd:sequence>
               <xsd:element name="item" type="xsd:string" minOccurs="0" maxOccurs="unbounded"/>
            </xsd:sequence>
         </xsd:complexType>

         <xsd:complexType name="Product">
            <xsd:sequence>
               <xsd:element name="name" type="xsd:string"/>
               <xsd:element name="display_name" type="xsd:string"/>
            </xsd:sequence>
         </xsd:complexType>
         <xsd:complexType name="Products">
            <xsd:sequence>
               <xsd:element name="product" type="types:Product" minOccurs="0" maxOccurs="unbounded"/>
            </xsd:sequence>
         </xsd:complexType>

         <xsd:complexType name="Topics">
            <xsd:sequence>
               <xsd:element name="topic" type="xsd:string" minOccurs="0" maxOccurs="unbounded"/>
            </xsd:sequence>
         </xsd:complexType>

         <xsd:complexType name="Model">
            <xsd:sequence>
               <xsd:element name="name" type="xsd:string"/>
               <xsd:element name="display_name" type="xsd:string"/>
               <xsd:element name="topics" type="types:Topics"/>
            </xsd:sequence>
         </xsd:complexType>
         <xsd:complexType name="Models">
            <xsd:sequence>
               <xsd:element name="model" type="types:Model" minOccurs="0" maxOccurs="unbounded"/>
            </xsd:sequence>
         </xsd:complexType>

         <xsd:complexType name="Option">
            <xsd:sequence>
               <xsd:element name="name" type="xsd:string"/>
               <xsd:element name="value" type="xsd:string"/>
```

```
            </xsd:sequence>
         </xsd:complexType>
         <xsd:complexType name="Options">
            <xsd:sequence>
               <xsd:element name="option" type="types:Option" minOccurs="0" maxOccurs="unbounded"/>
            </xsd:sequence>
         </xsd:complexType>

         <xsd:complexType name="SelectionFormat">
            <xsd:sequence>
               <xsd:element name="format" type="xsd:string"/>
               <xsd:element name="orientation" type="xsd:string"/>
               <xsd:element name="scales" type="xsd:string"/>
               <xsd:element name="format_defualt" type="xsd:string"/>
               <xsd:element name="orientation_default" type="xsd:string"/>
               <xsd:element name="scale_default" type="xsd:string"/>
            </xsd:sequence>
         </xsd:complexType>
         <xsd:complexType name="SelectionFormats">
            <xsd:sequence>
               <xsd:element name="selection_format" type="types:SeletionFormat" minOccurs="0" maxOc
            </xsd:sequence>
         </xsd:complexType>

         <xsd:complexType name="Params">
            <xsd:sequence>
               <xsd:element name="selection_type" type="xsd:string"/>
               <xsd:element name="selection_rotate" type="xsd:string"/>
               <xsd:element name="selection_formats" type="types:SelectionFormats"/>
               <xsd:element name="selection_options" type="types:Options"/>
               <xsd:element name="topic_selection_visible" type="xsd:string"/>
               <xsd:element name="order_la_options" type="types:Options"/>
               <xsd:element name="order_ra_option" type="xsd:string"/>
               <xsd:element name="order_ra_options" type="types:Options"/>
            </xsd:sequence>
         </xsd:complexType>

         <xsd:complexType name="ProductDefinition">
            <xsd:sequence>
               <xsd:element name="name" type="xsd:string"/>
               <xsd:element name="display_name" type="xsd:string"/>
               <xsd:element name="models" type="types:Models"/>
               <xsd:element name="params" type="types:Params"/>
               <xsd:element name="price_function" type="xsd:string"/>
            </xsd:sequence>
         </xsd:complexType>

      </xsd:schema>
   </types>

   <message name="getUser">
      <part name="user" type="xsd:string"/>
      <part name="password" type="xsd:string"/>
   </message>
   <message name="getUserResponse">
   </message>

   <message name="getProducts">
```

```
      <part name="user" type="xsd:string"/>
      <part name="password" type="xsd:string"/>
   </message>
   <message name="getProductsResponse">
      <part name="products" type="types:Products"/>
   </message>

   <message name="getProductDefinition">
      <part name="user" type="xsd:string"/>
      <part name="password" type="xsd:string"/>
      <part name="product" type="xsd:string"/>
   </message>
   <message name="getProductDefinitionResponse">
      <part name="productDefinition" type="types:ProductDefinition"/>
   </message>

   <portType name="UserServicePortType">
      <operation name="getUser">
         <input message="tns:getUserRequest"/>
         <output message="tns:getUserResponse"/>
      </operation>
      <operation name="getProducts">
         <input message="tns:getProductsRequest"/>
         <output message="tns:getProductsResponse"/>
      </operation>
      <operation name="getProductDefinition">
         <input message="tns:getProductDefinitionRequest"/>
         <output message="tns:getProductDefinitionResponse"/>
      </operation>
   </portType>

   <binding name="UserServiceBinding" type="tns:UserServicePortType">

      <soap:binding
         style="rpc"
         transport="http://schemas.xmlsoap.org/soap/http"/>

      <operation name="getUser">
         <soap:operation soapAction=""/>
         <input name="getUserRequest">
            <soap:body
               use="literal"
               namespace="http://www.infogrips.ch/geoshop/user/"/>
         </input>
         <output name="getUserResponse">
            <soap:body
               use="literal"
               namespace="http://www.infogrips.ch/geoshop/user/"/>
         </output>
      </operation>

      <operation name="getProduct">
         <soap:operation soapAction=""/>
         <input name="getProductRequest">
            <soap:body
               use="literal"
               namespace="http://www.infogrips.ch/geoshop/user/"/>
         </input>
```



```
         <output name="getProductResponse">
            <soap:body
               use="literal"
               namespace="http://www.infogrips.ch/geoshop/user/"/>
         </output>
      </operation>

      <operation name="getProductDefinition">
         <soap:operation soapAction=""/>
         <input name="getProductDefinitionRequest">
            <soap:body
               use="literal"
               namespace="http://www.infogrips.ch/geoshop/user/"/>
         </input>
         <output name="getProductDefinitionResponse">
            <soap:body
               use="literal"
               namespace="http://www.infogrips.ch/geoshop/user/"/>
         </output>
      </operation>

   </binding>

   <service name="UserService">
      <documentation>
         User service for GeoShop Server
      </documentation>
      <port name="GeoShopPort" binding="tns:UserServiceBinding">
         <soap:address location=
            "http://localhost:3501/soap/UserService.igs"
         />
      </port>
   </service>

</definitions>
```

B. OrderService WSDL Definitionen

```
<?xml version="1.0"?>
<definitions name="OrderService"

   targetNamespace="http://www.infogrips.ch/geoshop/order/"
   xmlns:tns="http://www.infogrips.ch/geoshop/order/"
   xmlns:types="http://www.infogrips.ch/geoshop/types/"
   xmlns:xsd="http://www.w3.org/2001/XMLSchema"
   xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
   xmlns="http://schemas.xmlsoap.org/wsdl/">
```

```
<types>
   <xsd:schema targetNamespace="http://www.infogrips.ch/geoshop/types/">
      <xsd:complexType name="StringList">
         <xsd:sequence>
            <xsd:element name="item" type="xsd:string"
                 minOccurs="0" maxOccurs="unbounded"/>
         </xsd:sequence>
      </xsd:complexType>
      <xsd:complexType name="Order">
         <xsd:sequence>
            <xsd:element name="orderno" type="xsd:int"/>
            <xsd:element name="orderurl" type="xsd:string"/>
            <xsd:element name="price" type="xsd:float"/>
            <xsd:element name="currency" type="xsd:string"/>
         </xsd:sequence>
      </xsd:complexType>
      <xsd:complexType name="Parameter">
         <xsd:sequence>
            <xsd:element name="name" type="xsd:string"/>
            <xsd:element name="value" type="xsd:string"/>
         </xsd:sequence>
      </xsd:complexType>
      <xsd:complexType name="Parameters">
         <xsd:sequence>
            <xsd:element name="parameter" type="types:Parameter"
                 minOccurs="0" maxOccurs="unbounded"/>
         </xsd:sequence>
      </xsd:complexType>
   </xsd:schema>
</types>

<message name="calculatePriceRequest">
   <part name="user" type="xsd:string"/>
   <part name="password" type="xsd:string"/>
   <part name="product" type="xsd:string"/>
   <part name="parameters" type="types:Parameters"/>
</message>
<message name="calculatePriceResponse">
   <part name="priceinfo" type="types:StringList"/>
</message>

<message name="sendOrderRequest">
   <part name="user" type="xsd:string"/>
   <part name="password" type="xsd:string"/>
   <part name="product" type="xsd:string"/>
   <part name="parameters" type="types:Parameters"/>
</message>
<message name="sendOrderResponse">
   <part name="order" type="types:Order"/>
</message>

<portType name="GeoShopPortType">
   <operation name="calculatePrice">
      <input message="tns:calculatePriceRequest"/>
      <output message="tns:calculatePriceResponse"/>
   </operation>
   <operation name="sendOrder">
      <input message="tns:sendOrderRequest"/>
```



```
         <output message="tns:sendOrderResponse"/>
      </operation>
   </portType>

   <binding name="GeoShopBinding" type="tns:GeoShopPortType">

      <soap:binding
         style="rpc"
         transport="http://schemas.xmlsoap.org/soap/http"/>

      <operation name="calculatePrice">
         <soap:operation soapAction=""/>
         <input name="calculatePriceRequest">
            <soap:body
               use="literal"
               namespace="http://www.infogrips.ch/geoshop/order/"/>
         </input>
         <output name="calculatePriceResponse">
            <soap:body
               use="literal"
               namespace="http://www.infogrips.ch/geoshop/order/"/>
         </output>
      </operation>

      <operation name="sendOrder">
         <soap:operation soapAction=""/>
         <input name="sendOrderRequest">
            <soap:body
               use="literal"
               namespace="http://www.infogrips.ch/geoshop/order/"/>
         </input>
         <output name="sendOrderResponse">
            <soap:body
               use="literal"
               namespace="http://www.infogrips.ch/geoshop/order/"/>
         </output>
      </operation>

   </binding>

   <service name="OrderService">
      <documentation>
         Price / Order service for GeoShop Server
      </documentation>
      <port name="GeoShopPort" binding="tns:GeoShopBinding">
         <soap:address location="http://localhost:3501/soap/OrderService.igs"
         />
      </port>
   </service>

</definitions>
```

C. TransformService WSDL Definitionen

```
<?xml version="1.0"?>
<definitions name="TransformService"

   targetNamespace="http://www.infogrips.ch/geoshop/transform/"
   xmlns:tns="http://www.infogrips.ch/geoshop/transform/"
   xmlns:types="http://www.infogrips.ch/geoshop/types/"
   xmlns:xsd="http://www.w3.org/2001/XMLSchema"
   xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
   xmlns="http://schemas.xmlsoap.org/wsdl/">

   <types>
      <xsd:schema targetNamespace="http://www.infogrips.ch/geoshop/types/">

         <xsd:complexType name="Point">
            <xsd:sequence>
               <xsd:element name="x" type="xsd:string"/>
               <xsd:element name="y" type="xsd:string"/>
               <xsd:element name="z" type="xsd:string"/>
            </xsd:sequence>
         </xsd:complexType>

      </xsd:schema>
   </types>

   <message name="getPointRequest">
      <part name="epsgin" type="xsd:string"/>
      <part name="epsgout" type="xsd:string"/>
      <part name="pointin" type="types:Point"/>
   </message>
   <message name="getPointResponse">
      <part name="pointout" type="types:Point"/>
   </message>

   <message name="getGeometryRequest">
      <part name="epsgin" type="xsd:string"/>
      <part name="epsgout" type="xsd:string"/>
      <part name="geometryin" type="xsd:string"/>
   </message>
   <message name="getGeometryResponse">
      <part name="geometryout" type="xsd:string"/>
   </message>

   <portType name="TransformServicePortType">
      <operation name="getPoint">
         <input message="tns:getPointRequest"/>
         <output message="tns:getPointResponse"/>
      </operation>
      <operation name="getGeometry">
         <input message="tns:getGeometryRequest"/>
         <output message="tns:getGeometryResponse"/>
      </operation>
```

```
    </portType>

    <binding name="TransformServiceBinding" type="tns:TransformServicePortType">

       <soap:binding
          style="rpc"
          transport="http://schemas.xmlsoap.org/soap/http"/>

       <operation name="getPoint">
          <soap:operation soapAction=""/>
          <input name="getPointRequest">
             <soap:body
                use="literal"
                namespace="http://www.infogrips.ch/geoshop/transform/"/>
          </input>
          <output name="getPointResponse">
             <soap:body
                use="literal"
                namespace="http://www.infogrips.ch/geoshop/transform/"/>
          </output>
       </operation>

       <operation name="getGeometry">
          <soap:operation soapAction=""/>
          <input name="getGeometryRequest">
             <soap:body
                use="literal"
                namespace="http://www.infogrips.ch/geoshop/transform/"/>
          </input>
          <output name="getGeometryResponse">
             <soap:body
                use="literal"
                namespace="http://www.infogrips.ch/geoshop/transform/"/>
          </output>
       </operation>

    </binding>

    <service name="TransformService">
       <documentation>
          Transform service for GeoShop Server
       </documentation>
       <port name="GeoShopPort" binding="tns:TransformServiceBinding">
          <soap:address location="http://localhost:3501/soap/TransformService.igs"/>
       </port>
    </service>

 </definitions>
```


D. OrderService Anwendungsbeispiel mit C#

Im nachfolgenden Beispiel soll eine einfache Bestellapplikation mit C# und Microsoft NET entwickelt werden. Dazu sind folgende Teilschritte notwendig:

1. Erstellen der C# Proxy Klasse für den Zugriff auf die GeoShop Methoden `calculatePrice` und `sendOrder` via WSDL (Webservice Definition Language).
2. Aufruf der Proxy Methoden mit C#.
3. Compilieren der Applikation.

1. Erstellen der C# Proxy Klasse via WSDL

Die C# Proxyklasse kann bequem mit dem Werkzeug `wsdl.exe` (Teil des Microsoft NET Framework SDK) generiert werden:

```
wsdl -language:CS -protocol:SOAP GEOSHOP_URL/soap/OrderClient.igs?wsdl
```

Für `GEOSHOP_URL` muss wieder die GeoShop Basisadresse eingesetzt werden. Als Resultat des Aufrufs wird die C# Datei `OrderClient.cs` generiert. `OrderClient.cs` enthält die C# Klasse `OrderClient` mit den beiden Methoden `calculatePrice` und `sendOrder`.

2. Aufruf der Proxy Methoden

Nachfolgend ist der Sourcecode der C# Bestellapplikation angegeben (Datei `OrderClient.cs`):

```
using System;

namespace soapclient
{

 class MainClass
 {

   public static void Main(string[] args)
   {

     OrderService os = new OrderService();

     String user = "test";
     String password = "test";
     String product = "dxf_dm01";

     // Die Bestellparameter setzen
     Parameter[] p = new Parameter[10];
     p[0] = new Parameter();
     p[0].name="min";
     p[0].value="675782.000/245373.000";
     p[1] = new Parameter();
     p[1].name="max";
     p[1].value="675839.000/245428.000";
```

```
      p[2] = new Parameter();
      p[2].name="model";
      p[2].value="DM01AVCH24D";
      p[3] = new Parameter();
      p[3].name="email";
      p[3].value="germann@infogrips.ch";
      p[4] = new Parameter();
      p[4].name="name1";
      p[4].value="infoGrips GmbH";
      p[5] = new Parameter();
      p[5].name="adr1";
      p[5].value="Obstgartenstr. 7";
      p[6] = new Parameter();
      p[6].name="zip";
      p[6].value="8035";
      p[7] = new Parameter();
      p[7].name="city";
      p[7].value="Zuerich";
      p[8] = new Parameter();
      p[8].name="verwendung";
      p[8].value="test";
      p[9] = new Parameter();
      p[9].name="tel";
      p[9].value="044 350 10 11";

      // Preisberechnung aufrufen
      Console.WriteLine("");
      Console.WriteLine("calculating price for product {0} ...",product);
      string[] priceinfo = os.calculatePrice(user,password,product,p);
      foreach (string s in priceinfo) {
        Console.WriteLine(s);
      }

      // Bestellung an den GeoShop abschicken
      Console.WriteLine("");
      Console.WriteLine("sending order to GeoShop Server ...");
      Order o = os.sendOrder(user,password,product,p);
      Console.WriteLine("orderno  is {0}",o.orderno);
      Console.WriteLine("orderurl is {0}",o.orderurl);
      Console.WriteLine("price    is {0}",o.price);
      Console.WriteLine("currency is {0}",o.currency);

    }

  }

}
```

3. Compilieren und Ausführen der Applikation

Die Applikation kann nun mit folgendem Befehl compiliert werden:

```
csc OrderService.cs OrderClient.cs
```

Der c# Compiler `csc.exe` erstellt als Resultat die Applikation `OrderClient.exe`, welche mit der Microsoft NET Runtime ausgeführt werden kann. Die Applikation `OrderClient.exe` generiert folgenden Output:

```
calculating price for product dxf_dm01 ...
40.0
SFr
0 ha          : SFr.     20.0
4 Ebenen      : SFr.     20.0
----------------------------
Zwischentotal : SFr.     40.0
----------------------------
Demorabatt    : SFr.    -40.0
Netto         : SFr.      0.0
MwSt. 7.6%    : SFr.      0.0
----------------------------
Total         : SFr.      0.0
============================

sending order to GeoShop Server ...
orderno  is 754
orderurl is http://localhost:3501/download/162111023650382/order.zip
price    is 40
currency is SFr
```